**University Mobile App For Equipment Loaning**

ISTM 6290: Final Project Report

Abdulmohsen Khashogji

Drithi Iyer

Mariamawit Anteneh

Ram Putrevu

# Table of Contents

[**Table of Contents**](#_pdovm7my1mre) **2**

[**Project Description**](#_xofdzr2ecqo9) **3**

[Business Problem](#_cpkq7utc7onp) 3

[Objective and Solution](#_i3e4ygkv8shm) 4

[**App Features**](#_ecruukhi323i) **4**

[Current Features](#_33srb718ehgj) 4

[Possible Future Improvements](#_onjm0r41o20l) 4

[**Usage**](#_y541fbmkvwrg) **4**

[Loan New Equipment:](#_z3ah3rfkmivd) 5

[Return Equipment:](#_wrberxrix1cv) 6

[**Design and Implementation Details**](#_b8i2j7l3mnka) **7**

[The Login Page](#_pqe6z8ct9p4q) 7

[Product Selection/Return Item Option](#_8jcdbvor002p) 8

[Equipment Options](#_cr097y7jo7dz) 8

[Checkout Form](#_stmrhvsfobsi) 9

[Return Form](#_qcx0yachz0hr) 9

[**Development Log**](#_bytzldxlqvyn) **10**

# 

# Project Description

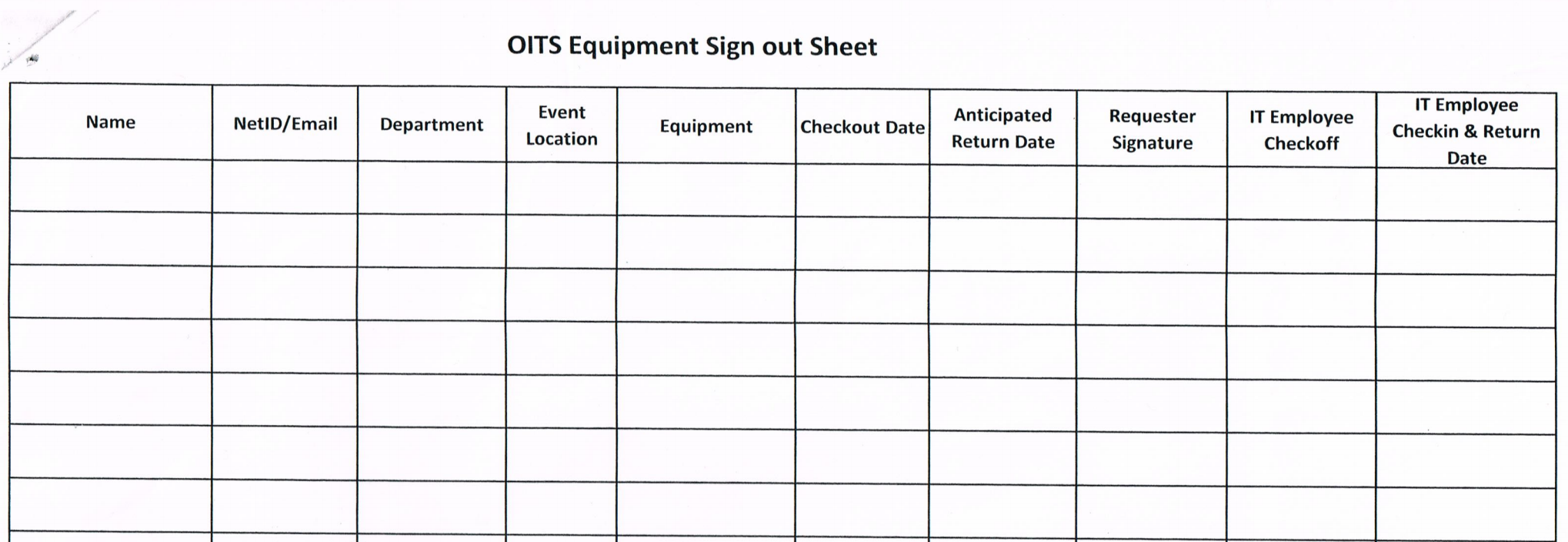
Foggy Bottom Campus Equipment Loaning System

Students and faculty can borrow equipment for a George Washington University (GWU) course at no additional cost. These equipments include audio recorders, cameras, laptops, video cameras, and clickers. The equipment may be retained for up to three business days.

## Business Problem

Currently, the department of Information Technology (IT) at the GWU School of Business loans out equipments to students and faculty members through a manual system that requires an authorized person to check off any equipment that are loaned. They do this on a paper-system to record different fields of the person checking out an equipment. When an equipment is returned, the authorized person must go back to look for the initial record and sign off in the “returned” column.

* Being a manual system, this system is not accurate in terms of real-time database updates to see the current inventory status.
* Since this is a paper-based system, there is a high risk for records to get misplaced and lost, thereby resulting in errors when an equipment is returned.
* A lot of time gets wasted for the authorized person to look for the initial record that was made in order to check-off an equipment that is being returned.



*Figure 1 - copy of the current paper system that is used by the IT department at GWSB*

## Objective and Solution

Our solution is to create a mobile application for students and faculty to check the availability of each equipment on a “real-time basis” and reserve it on the spot. The application will have forms to fill out any necessary details in order to validate the equipment to the user. The goal is to completely transform the current system in place to a more advanced system.

# App Features

## Current Features

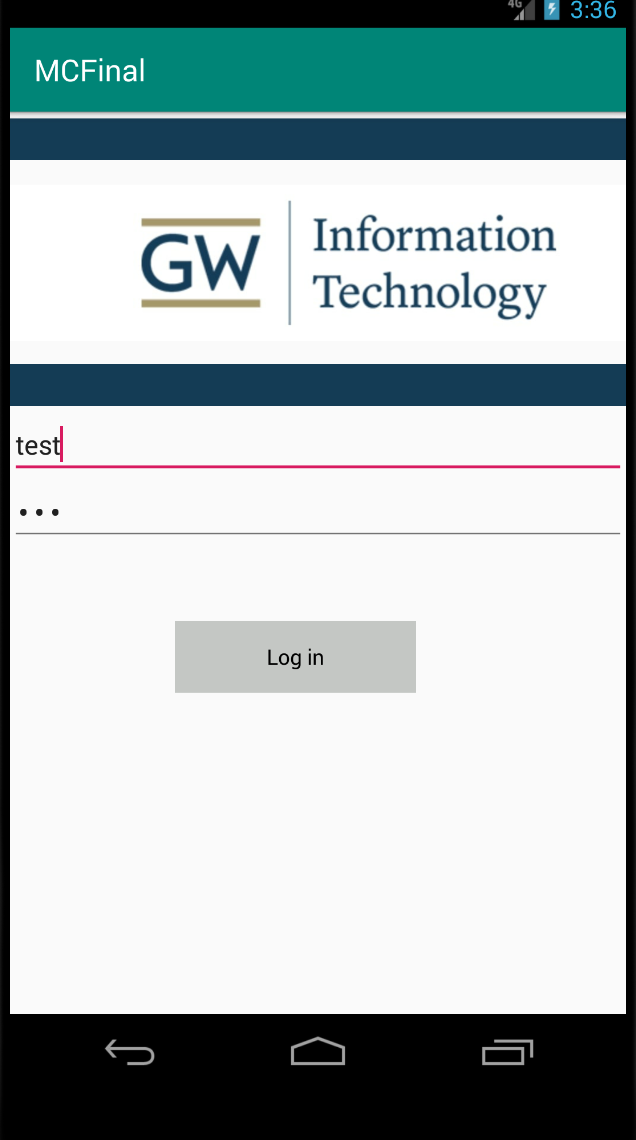
* When a user logs in, a session is created capturing the user’s information.
* The user can select from the categories mentioned (laptops, cameras, clickers etc.) and then choose an individual equipment they wish to checkout.
* To return, the user logs in to the portal to choose the equipment they want to return. The system then marks the equipment as “returned”, thereby allowing other users to reserve the equipment.

## Possible Future Improvements

* Adding functionalities such as charging for late returns and tracking damaged and lost equipment.
* Addition of a notification setting to remind users of product status and when the equipment is due for return.
* Updating user profile and history to view a log of equipment that have been checked out in the past, in case a user wants to reserve the same equipment.
* Finally, scaling this system to other schools in the university. Currently, the platform is only being tested in the School of Business at GWU.

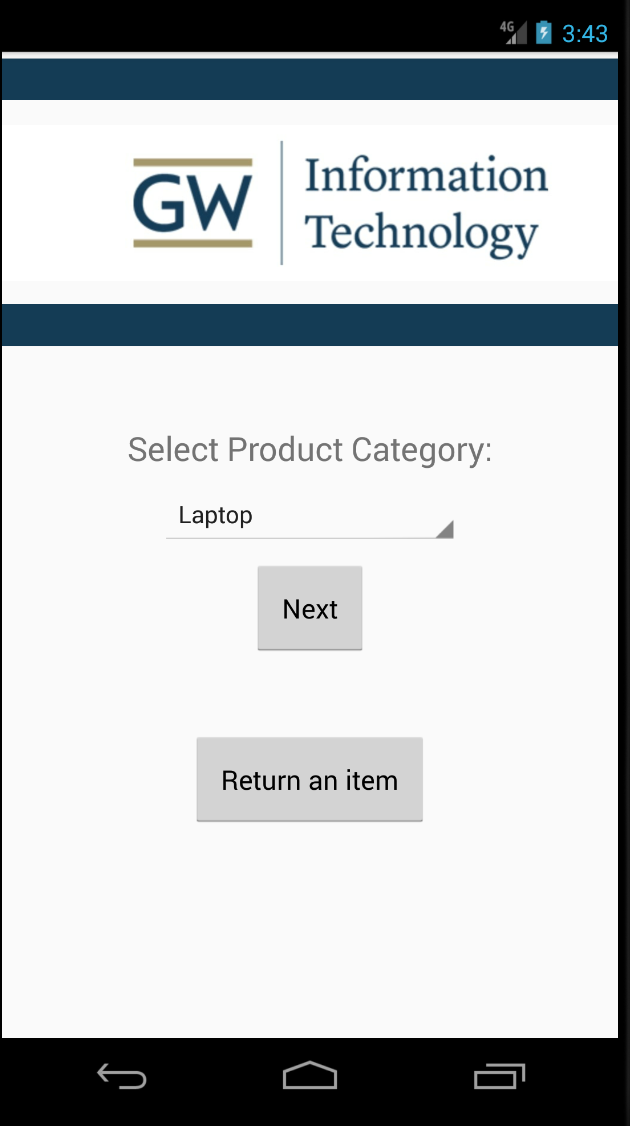
# Usage

In order to use the application, the user has to be a registered student with GWU School of Business. Then the user has to download the app from the store. Once you have downloaded the app, you will be seeing the screen as shown below. You will be using those credentials to log into the application. Enter your **NetID, Password** and click on **Log in** button.

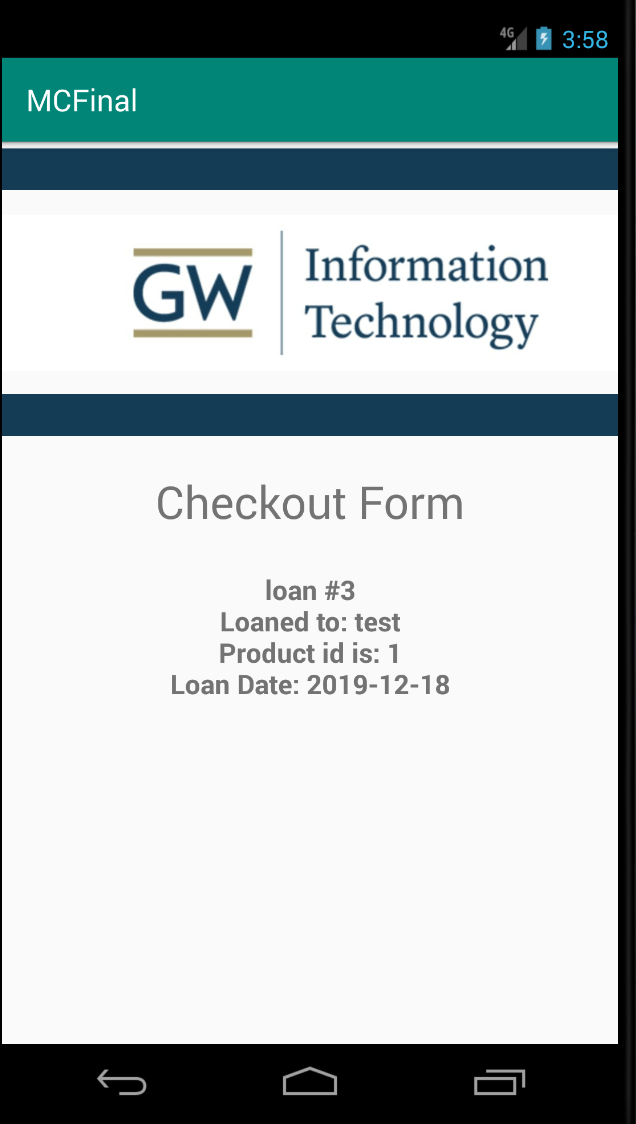
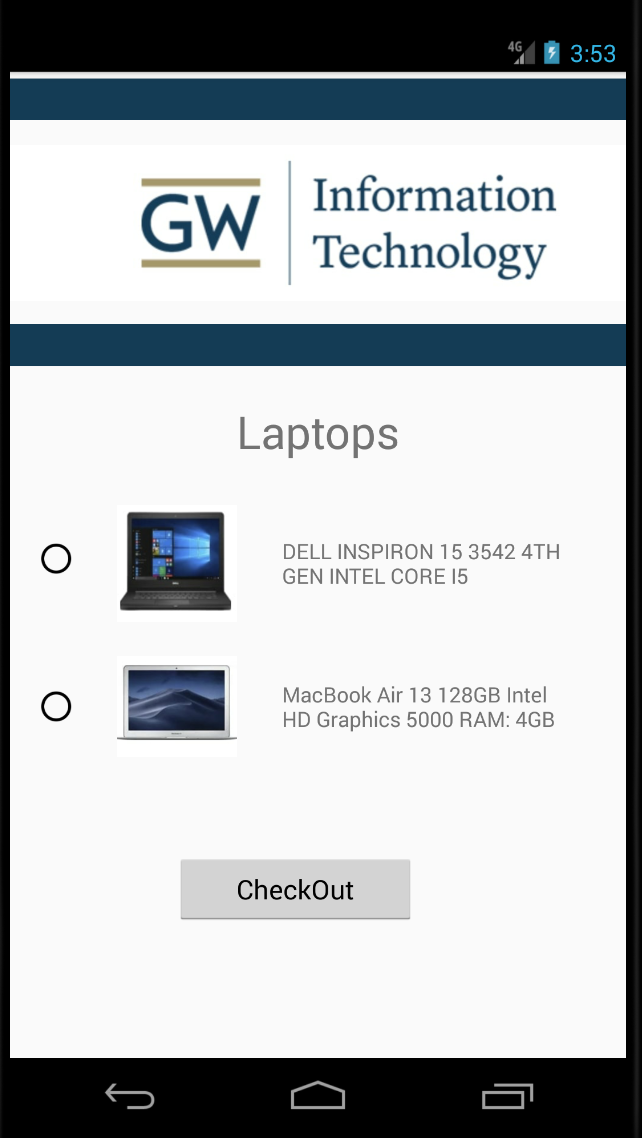


## Loan New Equipment:

**1.** You will be navigated to the page that has product category. You can select one category from a wide range of product categories from the drop-down menu. Then click on **Next** to view different items in the category page as shown below.



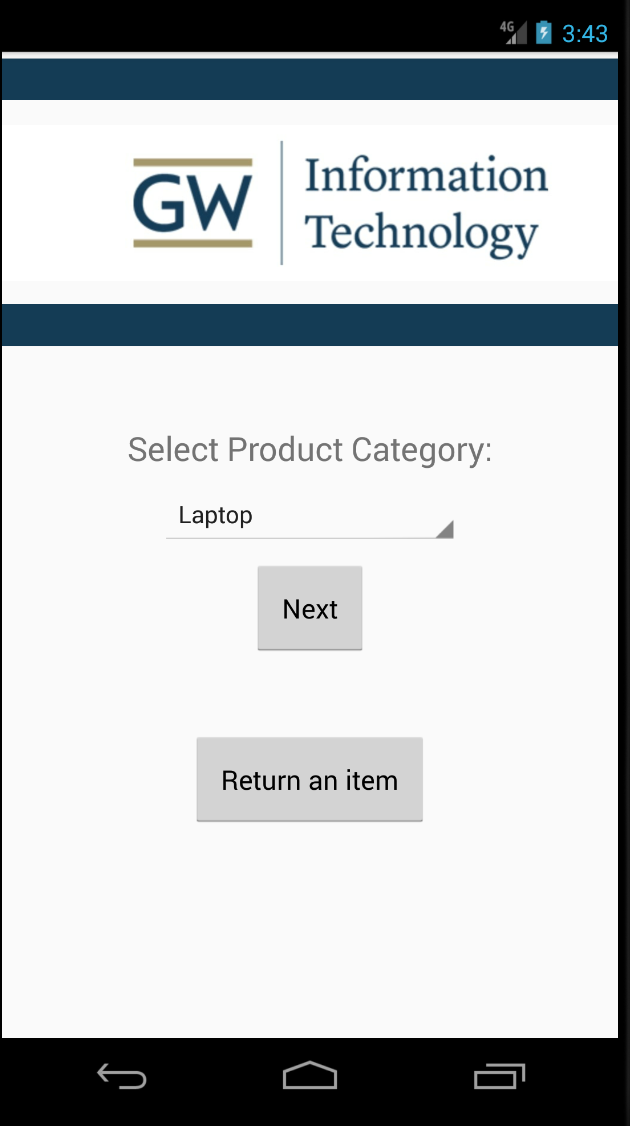
**2.** Select an item which you want to loan and click on the **checkout** button. Once you click on the button, the system will generate an electronic check out form request as shown below.



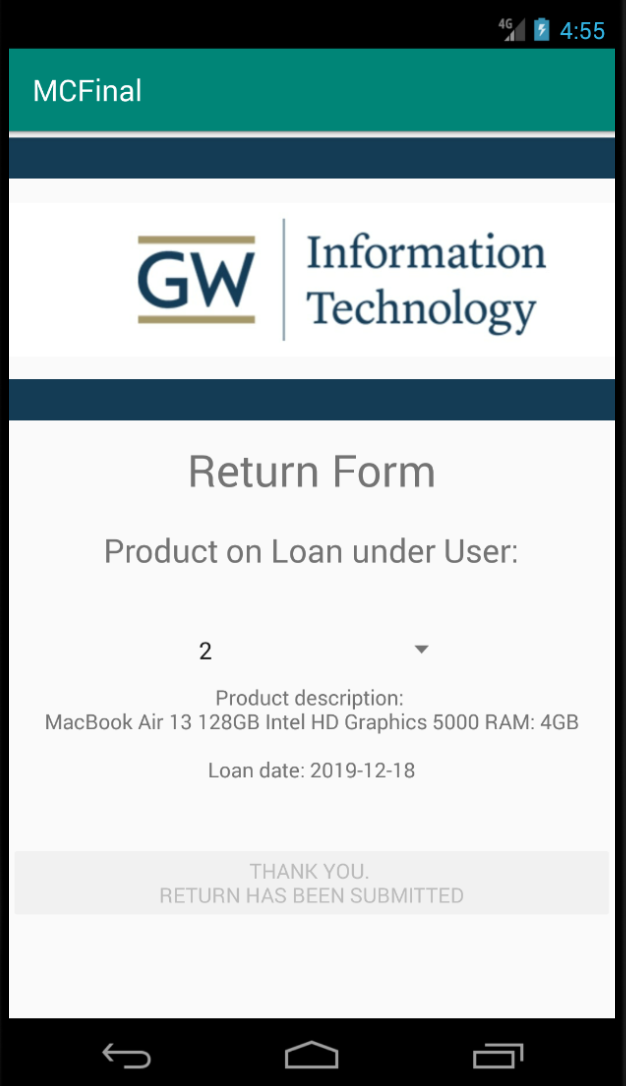
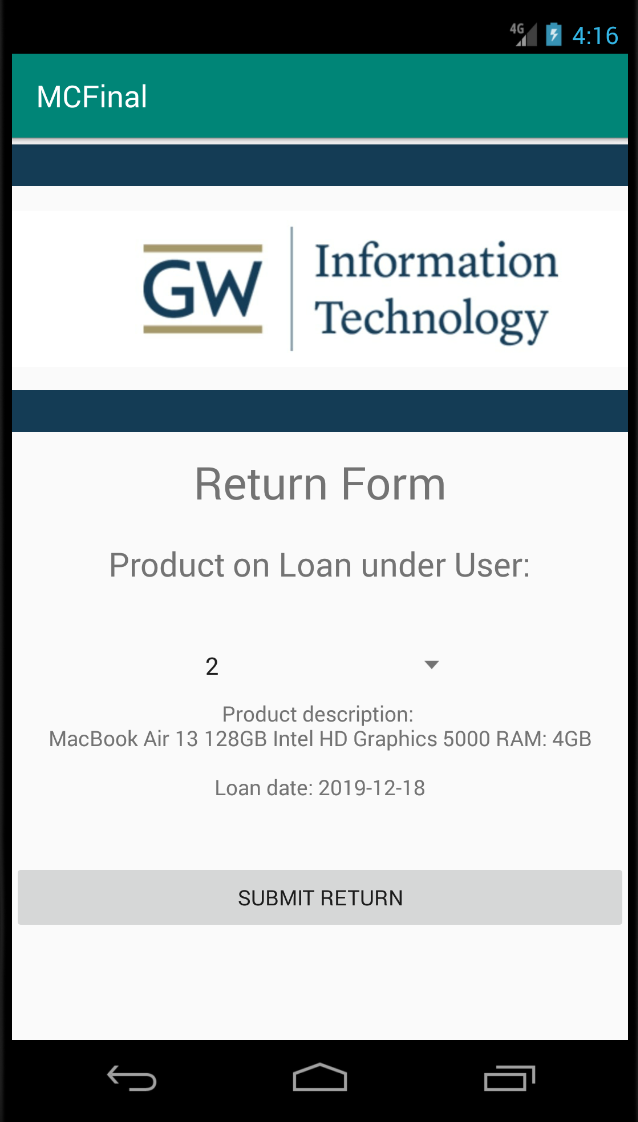
**3.** The System administrator will get notified, once the user submits the **checkout form**. Then, the administrator will reserve the equipment and the user will be able to pick up the equipment from the IT office.

## Return Equipment:

**1.**  Follow a similar procedure by logging into the application by using your **NetID** and **Password**. Then in the second page you need to click on **Return an item** button to navigate to return page.



**2.** Once you click the button, you will be seeing a **return form**, select **loan id** and click on **submit return** button. You will be submitting your return form and once you submit the form, you can return the equipment at the IT department.



# Design and Implementation Details

The design process for the app was to make it user friendly and easily understandable for all users. Since it is an application for the GWU community, we had some ideas for the design of the layout and what details it will entail. For demonstration purposes, we set out to limit the pages and functionalities of the application, however, with more resources and investment we believe it can become more advanced in the future. Currently, As we discussed above we have five layout pages developed and we will look at these pages in detail.

## The Login Page

In the xml file, using a linear layout method, the page design first has three image view buttons with the GWU IT department log and image color. Below there is a username and password text box where users can enter their information and be directed to the next page upon submission. The submission button is the Login button located right below the username and password. We used SQLite to create a database where data from users will be stored once they register for the account. Therefore when a user tries to login the program will check the database if the username and password exists. If it exists, the user will be logged in successfully, but if it doesn’t, it will set the text field of the password to null and show an error message using Toast.

## Product Selection/Return Item Option

The xml file for this page will have the first three images the login page did. Then we added a text view that instructs the user to ‘Select Product Category’. Below this instruction, we used the Spinner object so that the products that are available to choose can be shown when the user clicks the dropdown button. Once the user has chosen a product, he/she can click on the “Next” button found below where they would be directed to the next page. If the user has logged in to return an item borrowed at an earlier time, the button section below is used and leads to a different page. To do this we created instant of DatabaseHelper class to fetch the product type from the database. We then created ArrayList of strings to hold the Spinner content. The ArrayList will help us hold the type of products the user can check out. For this project purpose we only added laptop options in our database. After the user chooses an item, the program will take in the type of item chosen and will show the options available for that item on the next page. In addition, it passes on the information of the user from the previous screen (username and product type). The “return an item” button works in a similar way, when clicked it takes the user to the next page passing on the users information.

## Equipment Options

For this page, as demonstration we only added laptops in our database. Ideally, depending on what the user chooses to borrow from the previous page, the item type will vary and so will the options. The xml page similar to the first two, will have the three image views that show the school’s logo. The first part we see is the item type in a text view. Below we added the laptop type options using RadioButton in a RadioGroup so that a user is only able to one type of laptop from the option at a time. The buttons contain an image of the laptop and a detailed description of the type of laptop. In our code we included two types of laptops that are called from the database. Once the user has chosen, below there is a checkout button that when clicked will take the user to the checkout form. First the username and the product type will be saved to a string variable. Every item has a specific ID number, so depending on the choice of the user, the item will have an ID number; for example, in our app we only have two items, so If the user chooses Dell, the ID number will be set to 1. If Mac, then the ID will be set to 1. Before the product gets checked out, the program will check if the product is in stock. If it is, the user will be directed to the checkout page and the user’s information, product id and data will also be passed on. If the product is not available, it will display a notification saying there is no quantity available and choose another product.

## Checkout Form

In this page, the xml contains a checkout page that displays the checkout details of the user that is passed on from the previous page. There is a text view box that has “Checkout Form” and below there is a multi line text view that shows the details of the checkout depending on the user and the item chosen. First, we created a variable to determine the loan\_id. The variable fetches the latest (max) loan\_id from the database, if none is set, then it starts at 1, and increases as the user takes out more items. The number does not reset even if the user logs out and logs back in. Every time the user checks out an item, the quantity of the product that was checked out will decrease by 1. To confirm that the loan information is stored in the database, we display the information by getting the loan data from the database directly.

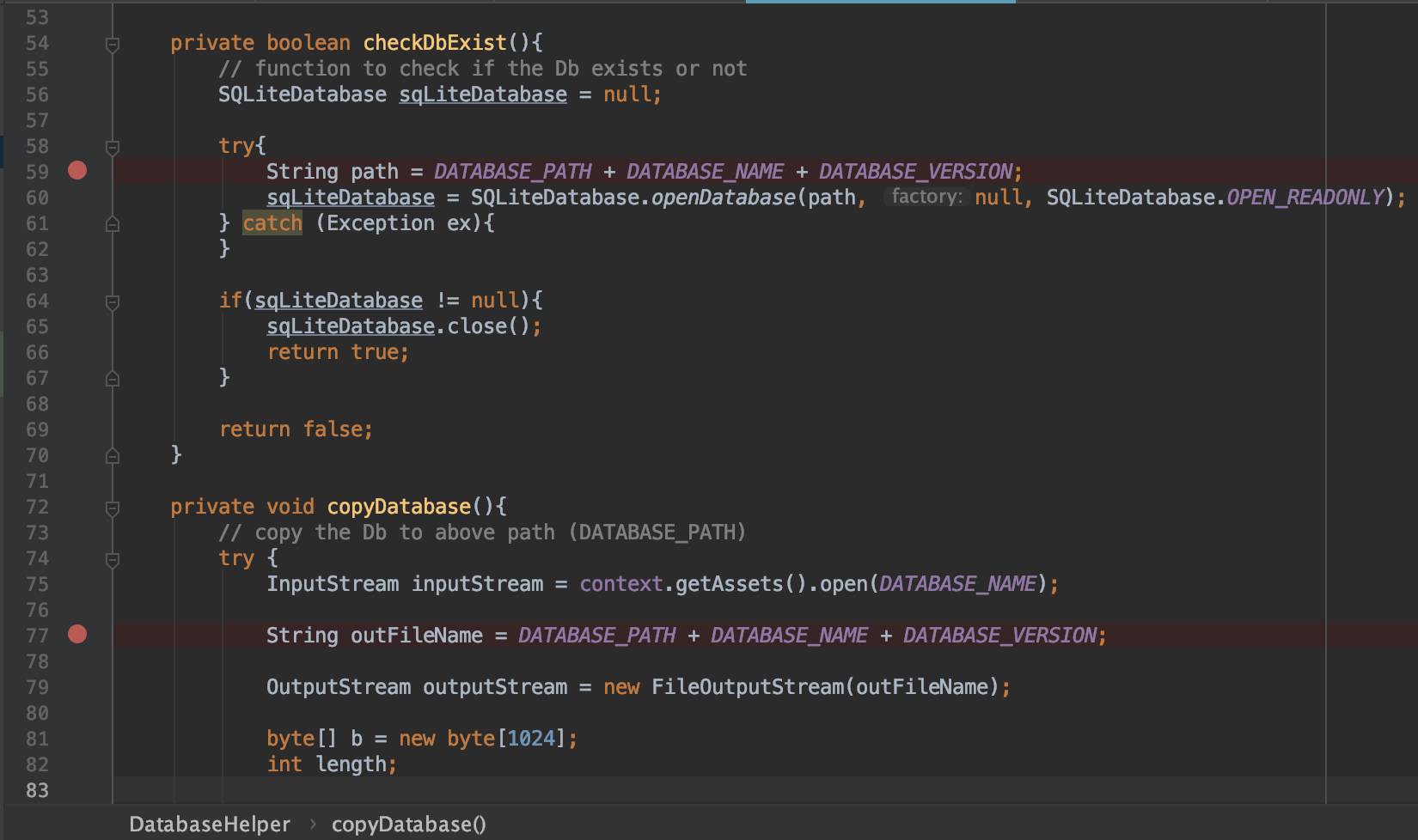
## Return Form

The xml layout for the return form having the same image view for the GWU IT as previous pages, first it has a text view for the “Return form” then below there is another text view “Product on loan under user” that directs the user to see how many products are already on loan under their account. The spinner is used here where the user can click and see the loan number (loan\_id) assigned that when he/she checked out. Below that is a multi-line text view that shows the product description for the item on loan. Lastly, we have the “Submit Return” button to finish the return process, when the user clicks on it, thank you message will appear in place of the “Submit Return” button. On the Java side, first, we create an instance of the database class to check the items in the database. Then if the user has not taken any items, it will not display any items and disable the submit button. Otherwise, it will check all open loans for the logged in user from the database and fetch the results to an array. When the item is returned successfully, it will display the description of the product and date. After that the loan id will be removed from the database.

# Development Log

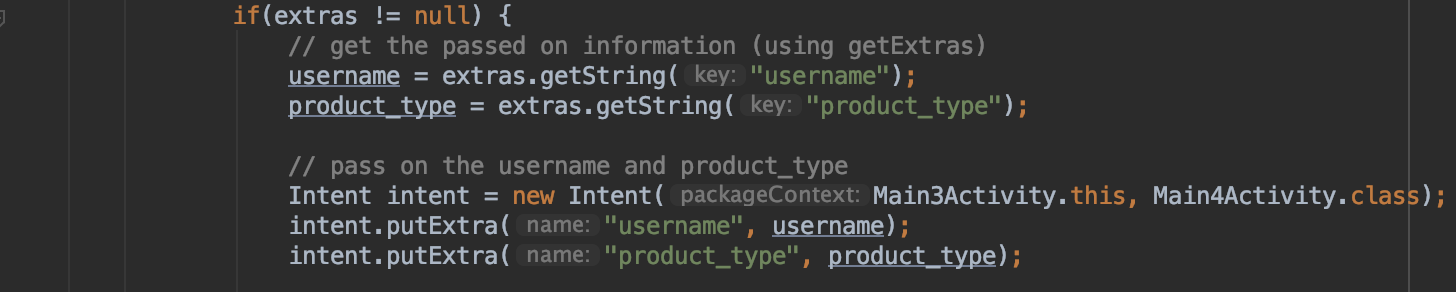
First, we implemented SQLite as our app database. We did that by designing the database and then adding the database file to our app’s assets folder (it\_db.db). From there, we read the database.

Our issue was once we update the database file, those updates are not reflected in our app, such as adding new products or new quantities. To trace the problem, we examined our DatabaseHelper class (in DatabaseHelper.java), we have a variable named DATABASE\_VERSION, which if increased should force the app to re-read the database. But that was not happening. To fix it, we modified the copyDatabase method to include the database version in the file name when we copy the database. Moreover, we modified the checkDbExist method to check if the current database version exists or not, by also including the version number in the database name. Both lines highlighted in red below:

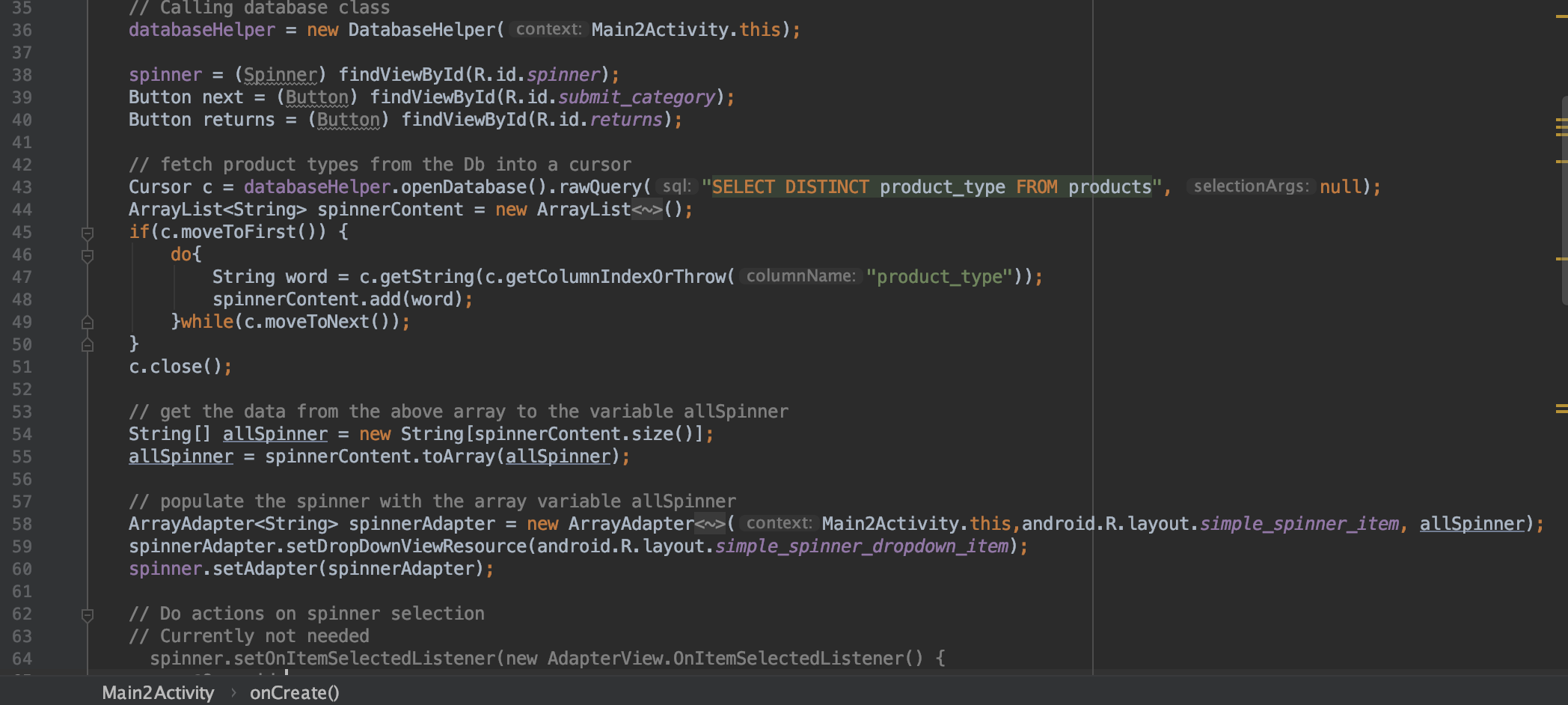


Second, we tried implementing Sign In with Google instead of the traditional username and password screen. We were successful in implementing it but there was one issue. We had to only accept email addresses that are from GWU’s addresses. That was difficult to achieve, thus, we reverted to use the simple login page. Where a user would have to enter the username and password, and that would be checked against the database.

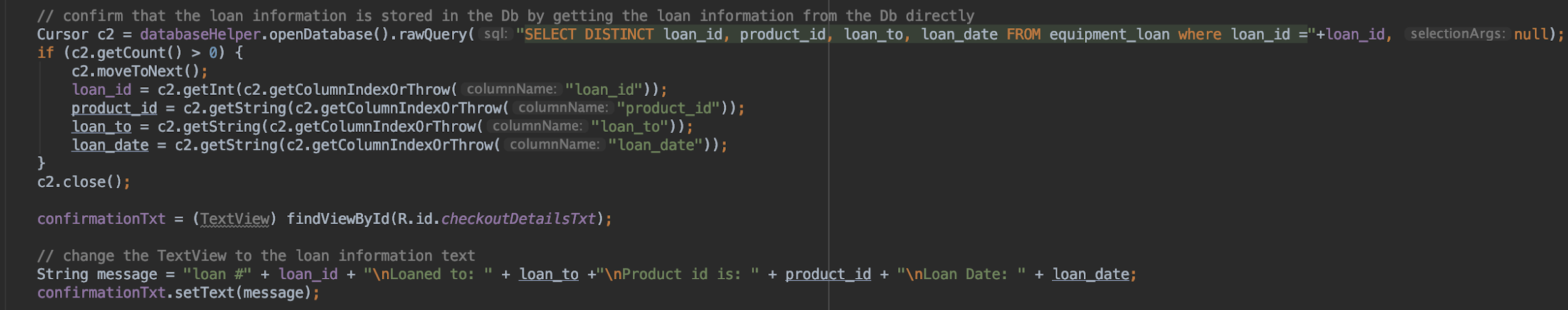
Third, how to pass information from one page to another? We had to do it from the first page, if the user successfully logins, then pass the username to the next page, then the product selected, and so on. We’re using Intent to move from one activity to another, and in Intent there are two functionalities that we can use to pass information and read it, putExtra() and readExtra(), respectively. Example:



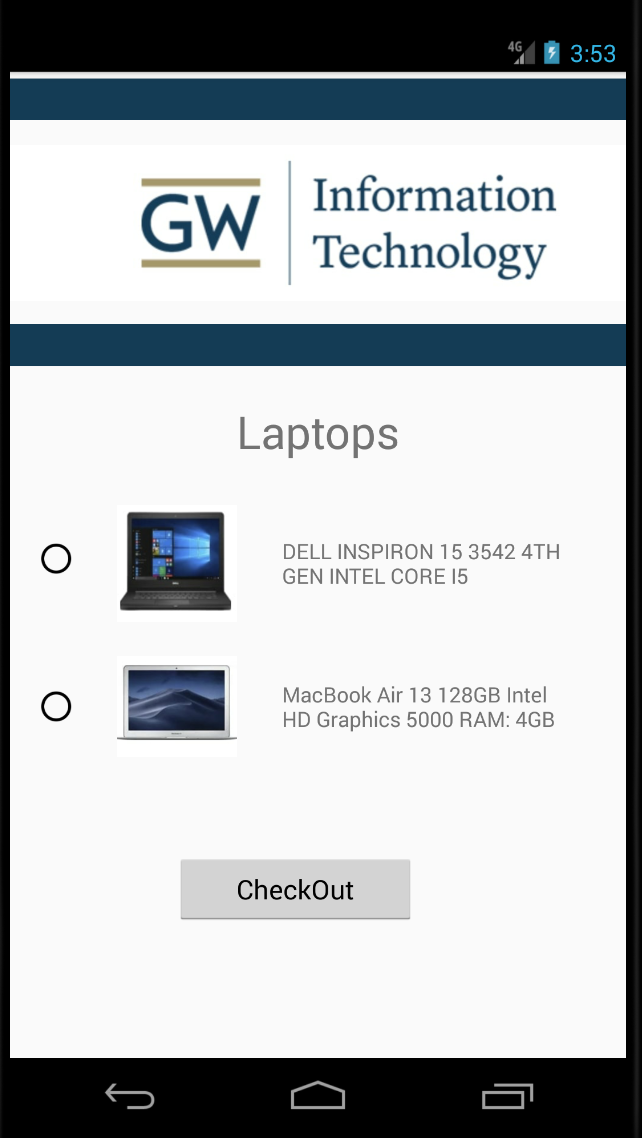
Fourth, we are using Spinners in two pages, that has to be populated from the database (dynamically), the product category selection and the returns form. To do so, we had to use Arrays, to load the data from the database and then set the Spinner to read using ArrayAdapter. Specifically, we used do while loop to read the data from the database, each time inserting the database values into an ArrayList, using the .add functionality. Then we created a variable that read all of the values of the ArrayList and converted into a string array. From there we created an ArrayAdapter that read the data from the string array and then populated the Spinner. The whole process took us a couple of trial runs to figure out the exact syntax and the way to do it.



Fifth, checkout and returns information had to be written in the database, and we had to verify the it was successfully written. In the normal process, once a new loan is given, then the quantity of that product should decrease, and vice-versa with returns. Initially, we did not verify if information was written to the database or not. While, testing the process we found out that the quantity was not decreasing/increasing. Therefore, we changed the checkout and returns pages to show the confirmation information from the database instead of passing the information from one activity to another (Intent). That way, if there were any issues in the code, the confirmation text will not be displayed and allowed us to trace it. In this example, our issue was in the SQL statement, it was not selecting/inserting the correct loan\_id.



Finally, we are using a static list in our product selection list:



As you can see, we are using radio button to select the product. Initially, we wanted to display this list dynamically from the database. The problem was we were not able to figure out which product was selected. Therefore, we elected to use a static list we each unique identifier for each radio button (Dell = 1, Mac = 2).